МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе №3

по дисциплине

Шаблоны проектирования ПО

РУКОВОДИТЕЛЬ:

_____ ___Жевнерчук Д. В.___

(подпись) (фамилия, и.,о.)

СТУДЕНТЫ: ____Дёмин Д. И.____

__Карпычева А. Ю.__

_____ ____Пигасин Д. А.___

(подпись) (фамилия, и.,о.)

_____18-ИВТ-1_____

(шифр группы)

Работа защищена «___» _____

С оценкой _____

Нижний Новгород 2019

Вариант 6

Реализуйте систему генерации шаблона html страницы по описанию:

1. Состава разделов (header, section1, ... , section_n, footer)
2. Структуры каждой секции (количество блоков div)

Добавьте опциональную возможность выделять определенный раздел, блок рамкой

Сгенерированный html код вводите в консоль.

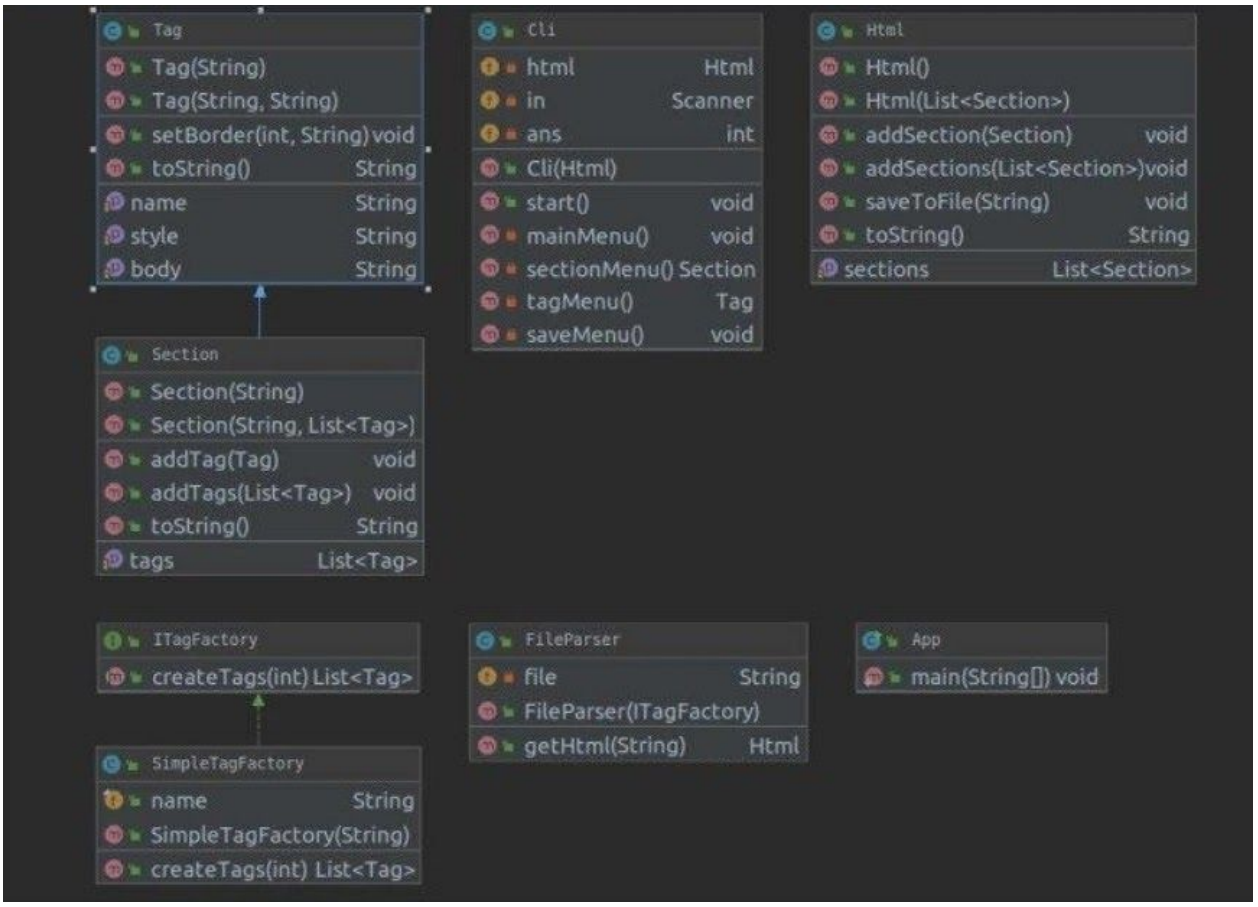
Выполнение

Программный код претерпел некоторые изменения: Функционал класса Section был перенесен в класс Tag.

Для выполнения работы были созданы классы ConfigurationGenerator и ConfigurationProcessor, описывающие потоки, генерирующие конфигурацию страницы и саму страницу соответственно. Также был создан разделяемый между ними ресурс RenderStack, содержащий массив конфигураций. Контроль доступа к критическим секциям производится с помощью механизма мониторов. Вывод сгенерированной разметки производится в консоль.

Реализация

UML схема



При написании программы мы руководствовались основными принципами ООП, а также использовали паттерны проектирования GOF там, где это уместно.

Например, конструктор класса FileParser принимает экземпляр фабрики, реализующей интерфейс ITagFactory.

Такой подход позволит нам изменять логику создания тэгов FileParser’ом, не переписывая то, что уже было сделано.

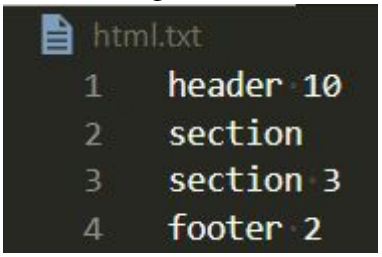
Для этого нужно описать новый класс, реализующий интерфейс ITagFactory и содержащий нужный нам алгоритм.

Как использовать

1. Описываем структуру страницы во внешнем файле.
    - Сначала указывается имя секции, количество вложенных блоков.
    - Одна секция - одна строка
2. Запускаем программу
3. Выделяем нужные блоки рамкой (опционально)
4. Сохраняем полученный шаблон (опционально)

Пример

1. Создаем файл html.txt:



2.Запускаем программу.

Код программы

app.App.java

```
package app;

import java.util.ArrayList;
import java.util.List;

import html.SimpleTagFactory;
import resources.RenderStack;
import threads.ConfigurationGenerator;

public class App {
	public static void main(String[] args) throws Exception {
	RenderStack rs = new RenderStack(10, new SimpleTagFactory("div"));
	List<Thread> threads = new ArrayList<>();

	for (int i = 0; i < 10; i++)
	threads.add(new Thread(new ConfigurationGenerator(rs, 300)));

	for (int i = 0; i < 5; i++)
	threads.add(new Thread(new threads.ConfigurationProcessor(rs, 500)));

	for (Thread thread : threads)
	thread.start();

	try {
	Thread.sleep(5000);
	} catch (InterruptedException e) {
	}
```

```
        for (Thread thread : threads)
        thread.interrupt();
        }

}
```

app.Cli.java

```
package app;

import java.io.IOException;
import java.util.Scanner;

import html.base.Html;
import html.base.Tag;

public class Cli {
        private Html html;
        private Scanner in = new Scanner(System.in);
        private int ans;

        public Cli(Html html) {
        this.html = html;
        }

        public void start() throws IOException {
        do {
        mainMenu();
        } while (ans != 3);

        saveMenu();
        in.close();
```

```
}

private void mainMenu() {
System.out.println(html);
System.out.println("1.Add section border\n2.Add tag border\n3.Exit");
System.out.print(">>>");
ans = in.nextInt();

if (ans == 1) {
Tag section = sectionMenu();
section.setBorder(5, "black");
}

if (ans == 2) {
Tag tag = tagMenu();
tag.setBorder(5, "black");
}
}

private Tag sectionMenu() {
Tag[] sections = html.getSections();

for (int i = 0; i < sections.length; i++) {
System.out.println(i + "." + sections[i].getName());
}

System.out.print(">>>");
return sections[in.nextInt()];
}

private Tag tagMenu() {
```

```
        Tag section = sectionMenu();
        Tag[] tags = section.getInnerTags();

        for (int i = 0; i < tags.length; i++) {
        System.out.println(i + "." + tags[i]);
        }

        System.out.print(">>>");
        return tags[in.nextInt()];
        }

        private void saveMenu() throws IOException {
        System.out.println("\nSave the file as index.html?\n1.Yes\n2.No");
        ans = in.nextInt();

        if (ans == 1)
        html.saveToFile("index.html");
        }

}
```

html.base.Tag.java

```
package html.base;

import java.util.LinkedList;
import java.util.List;

public class Tag {
        private String name;
        private String text;
        private String style;
        private List<Tag> innerTags = new LinkedList<Tag>();
```

```
public Tag(String name) {
this.name = name;
}

public Tag(String name, String text) {
this.name = name;
this.text = text;
}

public Tag(String name, List<Tag> innerTags) {
this.name = name;
this.innerTags = innerTags;
}

public String getName() {
return name;
}

public void setName(String name) {
this.name = name;
}

public String getText() {
return text;
}

public void setText(String text) {
this.text = text;
}
```

```
public String getStyle() {
return style;
}

public void setStyle(String style) {
this.style = style;
}

public void setBorder(int thickness, String color) {
setStyle("border:" + thickness + "px solid " + color);
}

public Tag[] getInnerTags() {
return innerTags.toArray(new Tag[innerTags.size()]);
}

public void setInnerTags(List<Tag> tags) {
this.innerTags = tags;
}

public void addInnerTag(Tag tag) {
innerTags.add(tag);
}

public void addInnerTags(List<Tag> tags) {
for (Tag tag : tags) {
addInnerTag(tag);
}
}

@Override
```

```
	public String toString() {
	String tagHead = "<" + name;
	String tagBody = "";
	String tagTail = "";

	if (text != null)
	tagBody += text;

	if (innerTags.size() != 0) {
	tagTail = "\n</" + name + ">";
	for (Tag tag : innerTags) {
		tagBody += "\n\t" + tag;
	}
	} else {
	tagTail = "</" + name + ">";
	}

	if (style != null)
	tagHead += " style='" + style + "'";

	tagHead += ">";

	return tagHead + tagBody + tagTail;
	}

}
```

html.base.Html.java

```
package html.base;

import java.io.FileWriter;
import java.io.IOException;
```

```
import java.util.LinkedList;
import java.util.List;

public class Html {
	private List<Tag> tags = new LinkedList<Tag>();

	public Html() {
	}

	public Html(List<Tag> tags) {
	this.tags = tags;
	}

	public Tag[] getSections() {
	return tags.toArray(new Tag[tags.size()]);
	}

	public void setSections(List<Tag> tags) {
	this.tags = tags;
	}

	public void addSection(Tag tag) {
	tags.add(tag);
	}

	public void addSections(List<Tag> tags) {
	for (Tag tag : tags) {
	addSection(tag);
	}
	}
```

```
	public void saveToFile(String fileName) throws IOException {
	FileWriter file = new FileWriter(fileName);
	file.write(toString());
	file.close();
	}

	@Override
	public String toString() {
	String html = "";
	for (Tag tag : tags) {
	if (tag.getInnerTags().length != 0)
		html += tag + "\n\n";
	else {
		html += tag + "\n";
	}
	}
	return html;
	}

}
```

html.FileParser.java

```
package html;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

import html.base.Html;
import html.base.Tag;
```

```
public class FileParser {
	private ITagFactory tagFactory;

	public FileParser(ITagFactory tagFactory) {
	this.tagFactory = tagFactory;
	}

	public Html getHtml(String file) {
	Scanner scan;
	try {
	scan = new Scanner(new FileReader(file));
	List<Tag> sections = new LinkedList<Tag>();

	while (scan.hasNextLine() & scan.hasNext()) {
		Scanner line = new Scanner(scan.nextLine());
		Tag section = new Tag(line.next());

		if (line.hasNextInt())
		section.addInnerTags(tagFactory.createTags(line.nextInt()));

		sections.add(section);
	}
	scan.close();

	return new Html(sections);

	} catch (FileNotFoundException e) {
	e.printStackTrace();

	return new Html();
```

```
        }
        }

}
```

html.ITagFactory.java

```
package html;

import java.util.List;

import html.base.Tag;

public interface ITagFactory {
        public List<Tag> createTags(int count);
}
```

html.SimpleTagFactory.java

```
package html;

import java.util.LinkedList;
import java.util.List;

import html.base.Tag;

public class SimpleTagFactory implements ITagFactory {
        final public String name;

        public SimpleTagFactory(String name) {
        this.name = name;
        }

        public List<Tag> createTags(int count) {
        List<Tag> tags = new LinkedList<Tag>();
```

```
        for (int i = 0; i < count; i++) {
        tags.add(new Tag(name, name + "_" + i));
        }
        return tags;
        }
}
```

resources.RenderStack.java

```
package resources;

import html.ITagFactory;
import html.base.Html;

public class RenderStack {
        private int[] stack;
        private int index = 0;
        private ITagFactory tagFactory;

        public RenderStack(int n, ITagFactory tagFactory) {
        stack = new int[n];
        this.tagFactory = tagFactory;
        }

        private void push(int numberOfTags) {
        stack[index] = numberOfTags;
        System.out.println("В очередь добавлена конфигурация " + index + " (" + stack[index]
+ ").");
        index++;
        }

        private void process() {
        System.out.println("Конфигурация " + (index - 1) + " (" + stack[index - 1] + ")
обработана.");
```

```
System.out.println(new Html(tagFactory.createTags(stack[index - 1])));
index--;
}

public synchronized void pushByThread(int numberOfTags) {
while (index == stack.length)
try {
        wait();
} catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        break;
}
if (index != stack.length) {
push(numberOfTags);
if (index == 1)
        notifyAll();
}
}

public synchronized void processByThread() {
while (index == 0)
try {
        wait();
} catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        break;
}
if (index != 0) {
process();
if (index == (stack.length - 1))
        notifyAll();
```

```
        }
        }
}
```

threads.ConfigurationGenerator.java

```
package threads;

import java.util.Random;

import resources.RenderStack;

public class ConfigurationGenerator implements Runnable {
        private RenderStack rs;
        private Random random = new Random();
        private int sleepTime;

        public ConfigurationGenerator(RenderStack rs, int sleepTime) {
        this.rs = rs;
        this.sleepTime = sleepTime;
        }

        @Override
        public void run() {
        while (!Thread.currentThread().isInterrupted()) {
        rs.pushByThread(random.nextInt(10) + 1);
        try {
                Thread.sleep(sleepTime);
        } catch (InterruptedException e) {
                break;
        }
        }
        System.out.println("Генератор конфигураций остановлен...");
```

```
        }
}
```

threads.ConfigurationProcessor.java

```
package threads;

import resources.RenderStack;

public class ConfigurationProcessor implements Runnable {
        private RenderStack rs;
        private int sleepTime;

        public ConfigurationProcessor(RenderStack rs, int sleepTime) {
        this.rs = rs;
        this.sleepTime = sleepTime;
        }

        @Override
        public void run() {
        while (!Thread.currentThread().isInterrupted()) {
        rs.processByThread();
        try {
                Thread.sleep(sleepTime);
        } catch (InterruptedException e) {
                break;
        }
        }
        System.out.println("Процессор конфигураций остановлен...");
        }
}
```